An Introduction to SusyFit

Ayan Paul
ERC Ideas: NPFlavour
INFN, Sezione di Roma. Roma, Italy.
Università di Roma - La Sapienza. Roma, Italy.

New Physics at Belle II. Karlsruhe. 25th Feb 2015.

by far the most difficult part of the project has been giving the code a name… we still stand undecided…

SusyFit is how it started, now it is more than just susy.

have suggestions? email me: apaul2@alumni.nd.edu

the program

- ✓ an analysis tool for direct and indirect observables
- ✓ comes with Bayesian Analysis Tool based on Markov Chain Monte Carlo
- ✓ SM and BSM arranged modularly for extraction of model based computations
- ✓ possibilities of adding user-generated models of new dynamics
- ✓ possibilities of adding user-defined observables
- ✓ possibilities of performing any choice of statistical analysis using the library
- ✓ a handy tool for getting very quick (statistical) estimates and doing full-fledged statistical analyses
- ✓ deployable both on clusters and multicore CPUs for large statistical analyses.
- ✓ equally friendly for all level of users and developers (doxygened in detail)

the philosophy

everyone gets a candy they like

- we offer a variety of interfaces that can cater to beginners, advanced users and developers
- a variety of NP models and observables will be included and the developers can add more

statistical precision requires large samples

- a lot of focus has been put on speed with extensive caching built in
- built-in MPI parallelization for deployment on large clusters

open source and open for customization

- source will be released under GPL with extensive documentation
- working developer version always available through git (requires NetBeans IDE)

the dependencies

ROOT (https://root.cern.ch)

- it is used to plot and store all the histograms generated at run-time (in *.pdf, *.ps and *.root)
- for now we are using ROOT v5.xx
- ROOT v6.xx migration scheduled for later this year

BOOST (http://www.boost.org/)

- a largely header based implementation of efficient and safe memory accessing procedures and file parser in c++
- we are using the headers only so that building of boost is unnecessary

GSL (http://www.gnu.org/software/gsl/)

- the GNU Scientific Libraries: efficient matrix operations and integrals
- we have our own wrappers for GSL to aid future developers

the optional dependencies

BAT (https://www.mppmu.mpg.de/bat/)

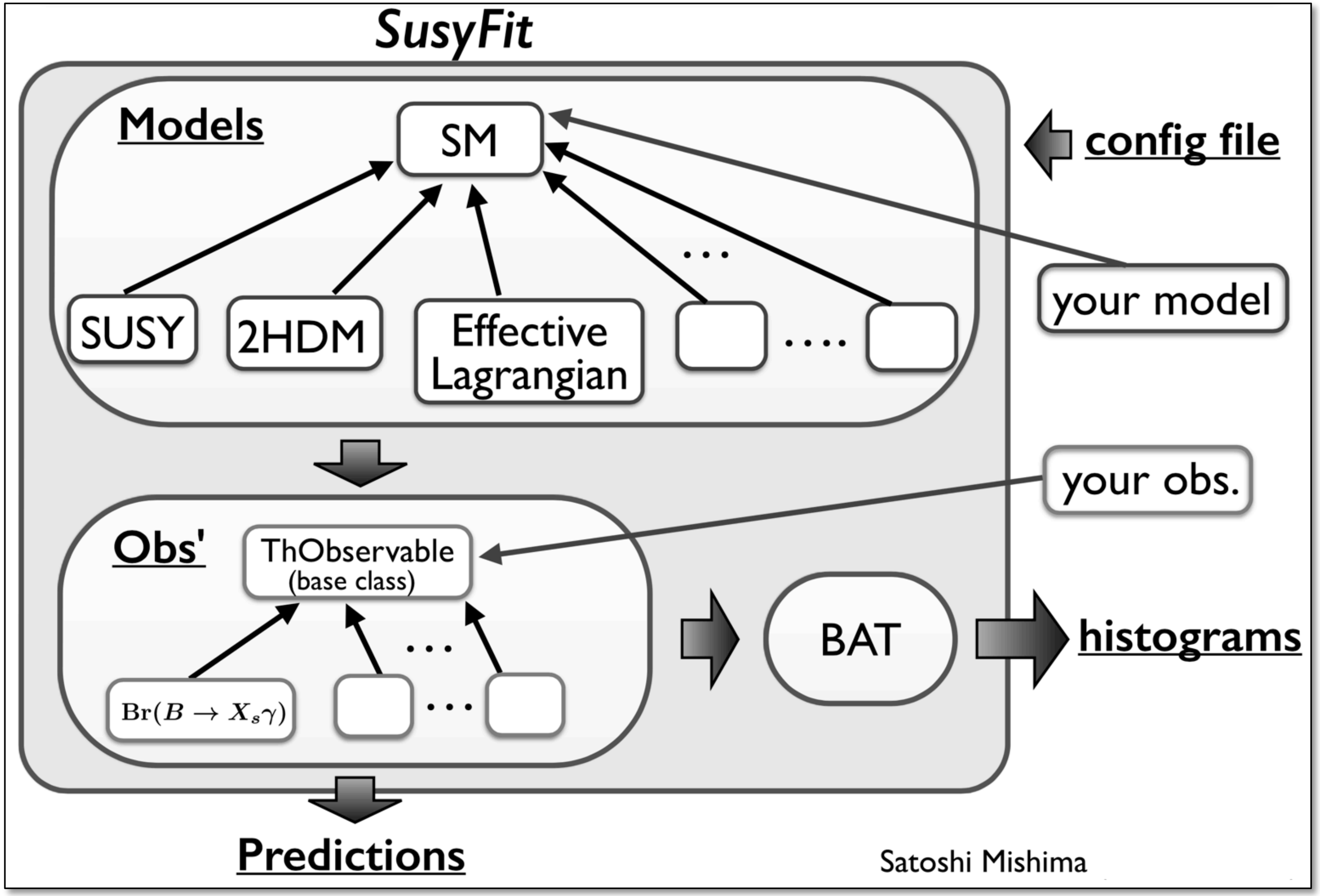
- necessary if our MCMC engine is used
- Bayesian Analysis Tool: developed separately as a code for Bayesian analysis based on Markov Chain Monte Carlo routines
- highly tunable in its procedures with different optimizing algorithms
- we provide access to some tuning parameters, others accessible through modification of our Monte Carlo engine
- currently compatible with BAT v0.9.3
- migration to BAT v0.9.4 scheduled for later this year

openMPI (http://www.open-mpi.org/) / MPICH (http://www.mpich.org/)

- necessary only for parallelized runs
- requires our patched version of BAT v0.9.3
- tested for large scale deployment @ O(10^3) cores in batch submission systems

got stuck with installation? email me: apaul2@alumni.nd.edu

the structure of the code



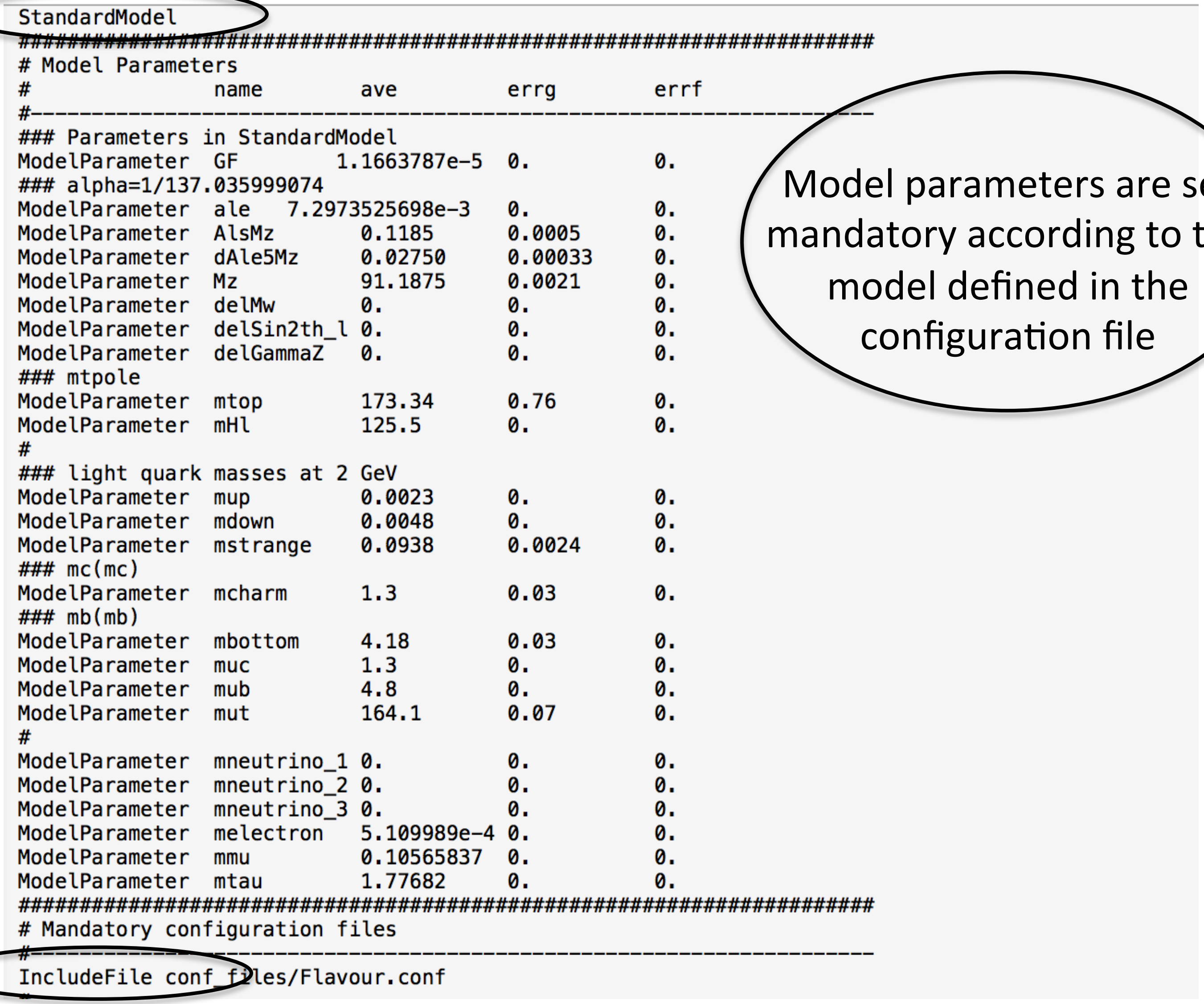
the structure of the code

all model parameters set by the user through a configuration files

Model definition

Model parameters are set mandatory according to the model defined in the configuration file

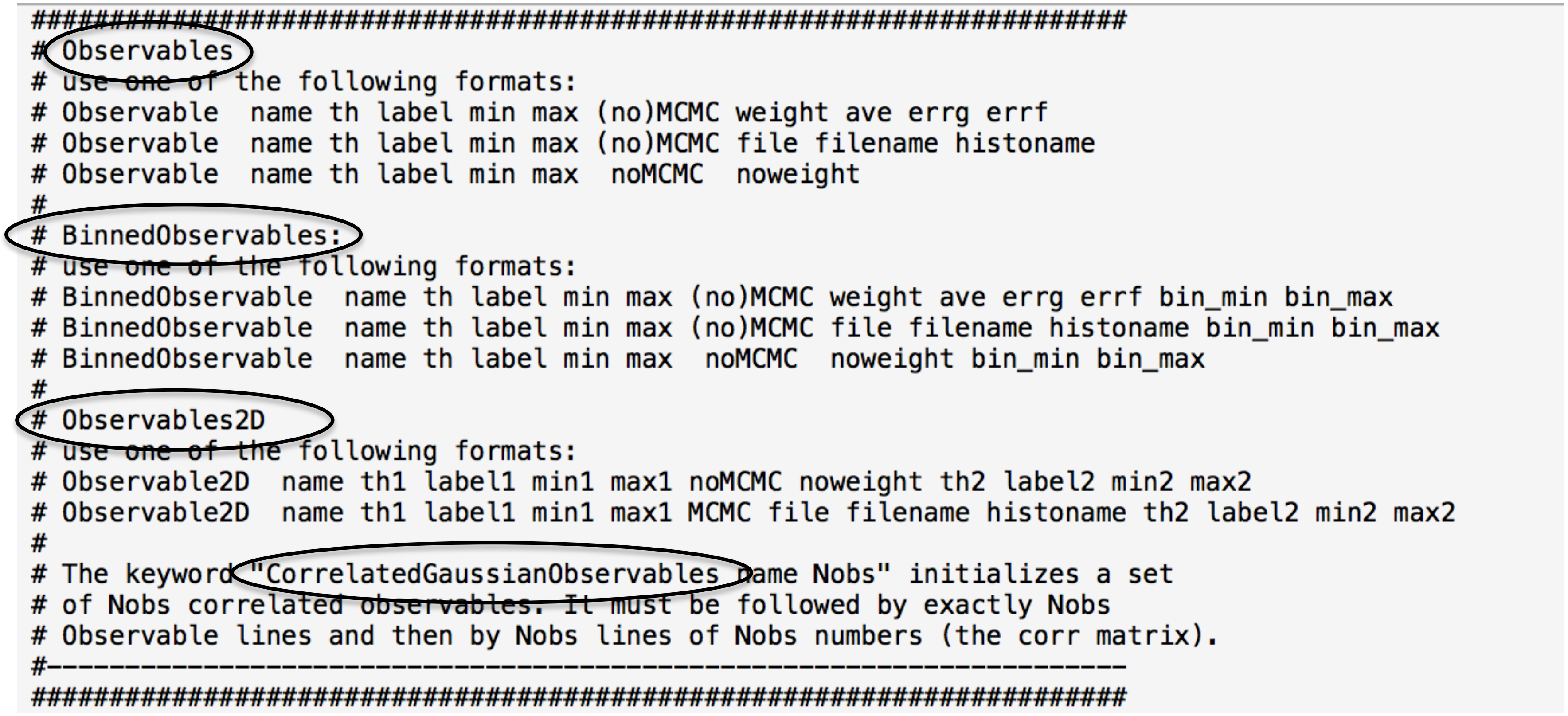
additional mandatory/ optional configuration files



```
StandardModel
######################################################################
# Model Parameters
#               name        ave         errg        errf
#---------------------------------------------------------------------
### Parameters in StandardModel
ModelParameter  GF       1.1663787e-5  0.          0.
### alpha=1/137.035999074
ModelParameter  ale   7.2973525698e-3  0.          0.
ModelParameter  AlsMz       0.1185      0.0005      0.
ModelParameter  dAle5Mz     0.02750     0.00033     0.
ModelParameter  Mz          91.1875     0.0021      0.
ModelParameter  delMw       0.          0.          0.
ModelParameter  delSin2th_l 0.          0.          0.
ModelParameter  delGammaZ   0.          0.          0.
### mtpole
ModelParameter  mtop        173.34      0.76        0.
ModelParameter  mHl         125.5       0.          0.
#
### light quark masses at 2 GeV
ModelParameter  mup         0.0023      0.          0.
ModelParameter  mdown       0.0048      0.          0.
ModelParameter  mstrange    0.0938      0.0024      0.
### mc(mc)
ModelParameter  mcharm      1.3         0.03        0.
### mb(mb)
ModelParameter  mbottom     4.18        0.03        0.
ModelParameter  muc         1.3         0.          0.
ModelParameter  mub         4.8         0.          0.
ModelParameter  mut         164.1       0.07        0.
#
ModelParameter  mneutrino_1 0.          0.          0.
ModelParameter  mneutrino_2 0.          0.          0.
ModelParameter  mneutrino_3 0.          0.          0.
ModelParameter  melectron   5.109989e-4 0.          0.
ModelParameter  mmu         0.10565837  0.          0.
ModelParameter  mtau        1.77682     0.          0.
######################################################################
# Mandatory configuration files
#---------------------------------------------------------------------
IncludeFile conf_files/Flavour.conf
#
```

the structure of the code

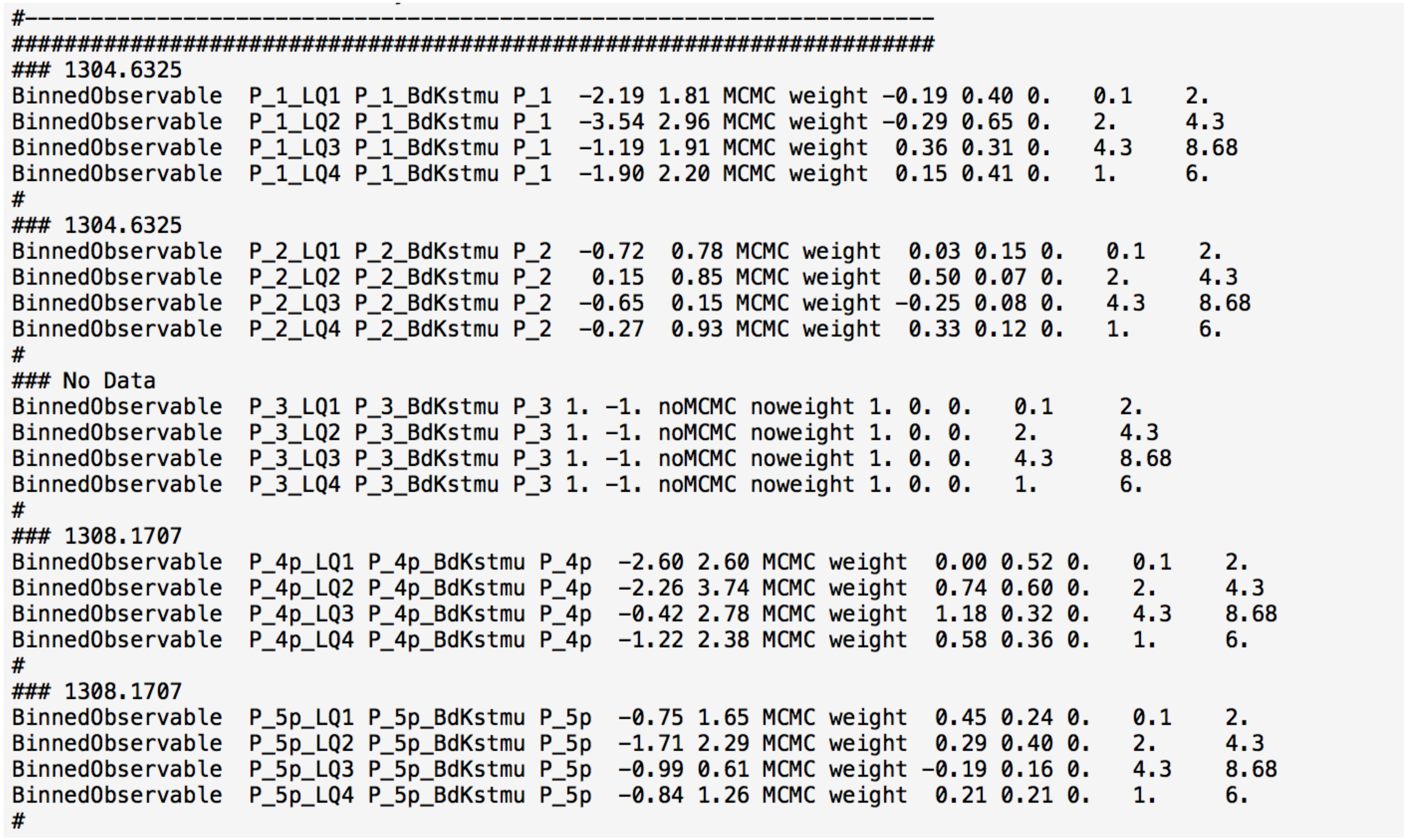
observables list set by user in the configuration files



```
######################################################################
# Observables
# use one of the following formats:
# Observable  name th label min max (no)MCMC weight ave errg errf
# Observable  name th label min max (no)MCMC file filename histoname
# Observable  name th label min max  noMCMC  noweight
#
# BinnedObservables:
# use one of the following formats:
# BinnedObservable  name th label min max (no)MCMC weight ave errg errf bin_min bin_max
# BinnedObservable  name th label min max (no)MCMC file filename histoname bin_min bin_max
# BinnedObservable  name th label min max  noMCMC  noweight bin_min bin_max
#
# Observables2D
# use one of the following formats:
# Observable2D  name th1 label1 min1 max1 noMCMC noweight th2 label2 min2 max2
# Observable2D  name th1 label1 min1 max1 MCMC file filename histoname th2 label2 min2 max2
#
# The keyword "CorrelatedGaussianObservables name Nobs" initializes a set
# of Nobs correlated observables. It must be followed by exactly Nobs
# Observable lines and then by Nobs lines of Nobs numbers (the corr matrix).
#----------------------------------------------------------------------
######################################################################
```

the structure of the code

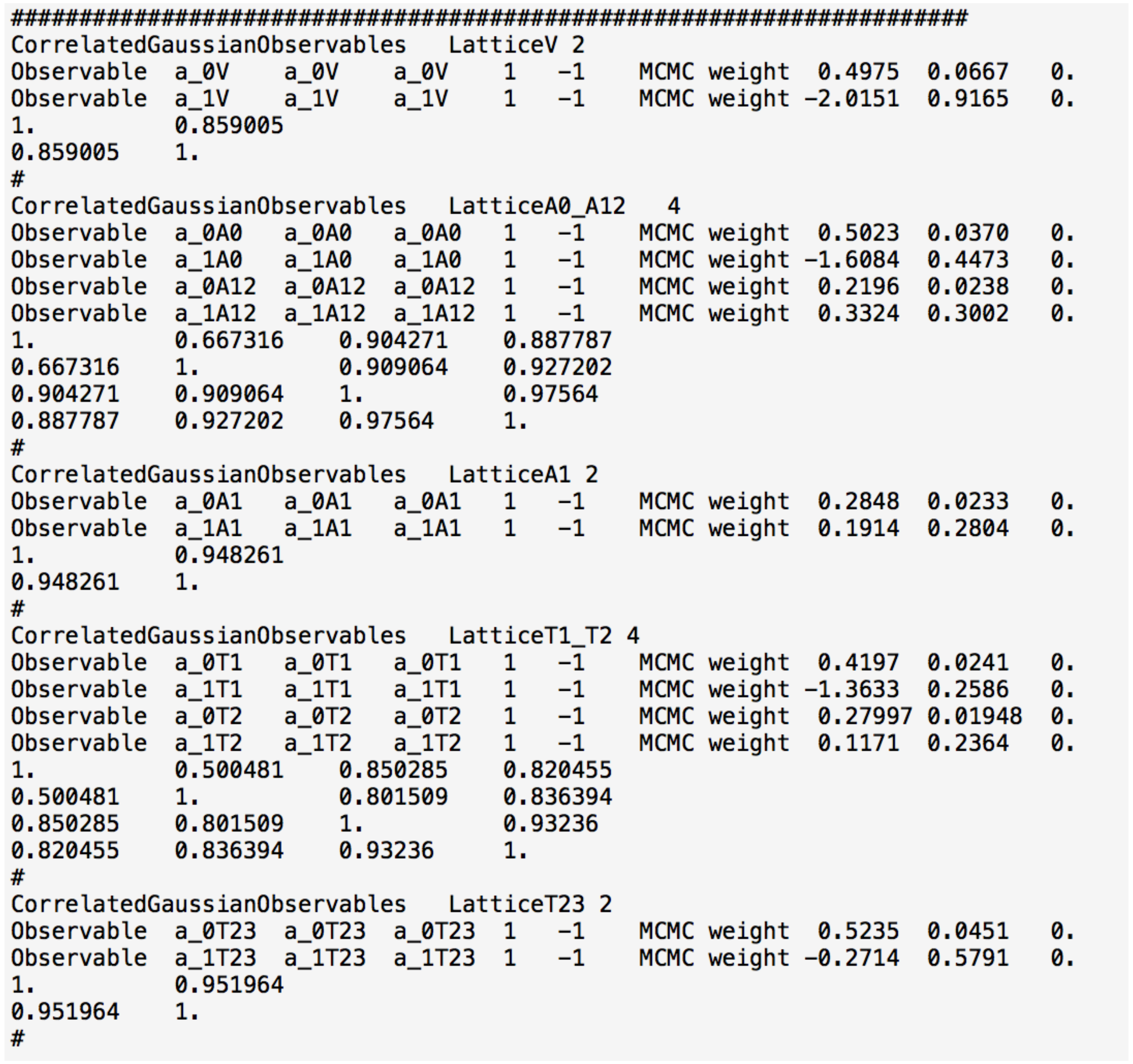
observables list set by user in the configuration files



```
#----------------------------------------------------------------------
######################################################################
### 1304.6325
BinnedObservable  P_1_LQ1 P_1_BdKstmu P_1  -2.19 1.81 MCMC weight -0.19 0.40 0.   0.1    2.
BinnedObservable  P_1_LQ2 P_1_BdKstmu P_1  -3.54 2.96 MCMC weight -0.29 0.65 0.   2.     4.3
BinnedObservable  P_1_LQ3 P_1_BdKstmu P_1  -1.19 1.91 MCMC weight  0.36 0.31 0.   4.3    8.68
BinnedObservable  P_1_LQ4 P_1_BdKstmu P_1  -1.90 2.20 MCMC weight  0.15 0.41 0.   1.     6.
#
### 1304.6325
BinnedObservable  P_2_LQ1 P_2_BdKstmu P_2  -0.72  0.78 MCMC weight  0.03 0.15 0.   0.1    2.
BinnedObservable  P_2_LQ2 P_2_BdKstmu P_2   0.15  0.85 MCMC weight  0.50 0.07 0.   2.     4.3
BinnedObservable  P_2_LQ3 P_2_BdKstmu P_2  -0.65  0.15 MCMC weight -0.25 0.08 0.   4.3    8.68
BinnedObservable  P_2_LQ4 P_2_BdKstmu P_2  -0.27  0.93 MCMC weight  0.33 0.12 0.   1.     6.
#
### No Data
BinnedObservable  P_3_LQ1 P_3_BdKstmu P_3 1. -1. noMCMC noweight 1. 0. 0.   0.1     2.
BinnedObservable  P_3_LQ2 P_3_BdKstmu P_3 1. -1. noMCMC noweight 1. 0. 0.   2.      4.3
BinnedObservable  P_3_LQ3 P_3_BdKstmu P_3 1. -1. noMCMC noweight 1. 0. 0.   4.3     8.68
BinnedObservable  P_3_LQ4 P_3_BdKstmu P_3 1. -1. noMCMC noweight 1. 0. 0.   1.      6.
#
### 1308.1707
BinnedObservable  P_4p_LQ1 P_4p_BdKstmu P_4p  -2.60 2.60 MCMC weight  0.00 0.52 0.   0.1    2.
BinnedObservable  P_4p_LQ2 P_4p_BdKstmu P_4p  -2.26 3.74 MCMC weight  0.74 0.60 0.   2.     4.3
BinnedObservable  P_4p_LQ3 P_4p_BdKstmu P_4p  -0.42 2.78 MCMC weight  1.18 0.32 0.   4.3    8.68
BinnedObservable  P_4p_LQ4 P_4p_BdKstmu P_4p  -1.22 2.38 MCMC weight  0.58 0.36 0.   1.     6.
#
### 1308.1707
BinnedObservable  P_5p_LQ1 P_5p_BdKstmu P_5p  -0.75 1.65 MCMC weight  0.45 0.24 0.   0.1    2.
BinnedObservable  P_5p_LQ2 P_5p_BdKstmu P_5p  -1.71 2.29 MCMC weight  0.29 0.40 0.   2.     4.3
BinnedObservable  P_5p_LQ3 P_5p_BdKstmu P_5p  -0.99 0.61 MCMC weight -0.19 0.16 0.   4.3    8.68
BinnedObservable  P_5p_LQ4 P_5p_BdKstmu P_5p  -0.84 1.26 MCMC weight  0.21 0.21 0.   1.     6.
#
```

the structure of the code

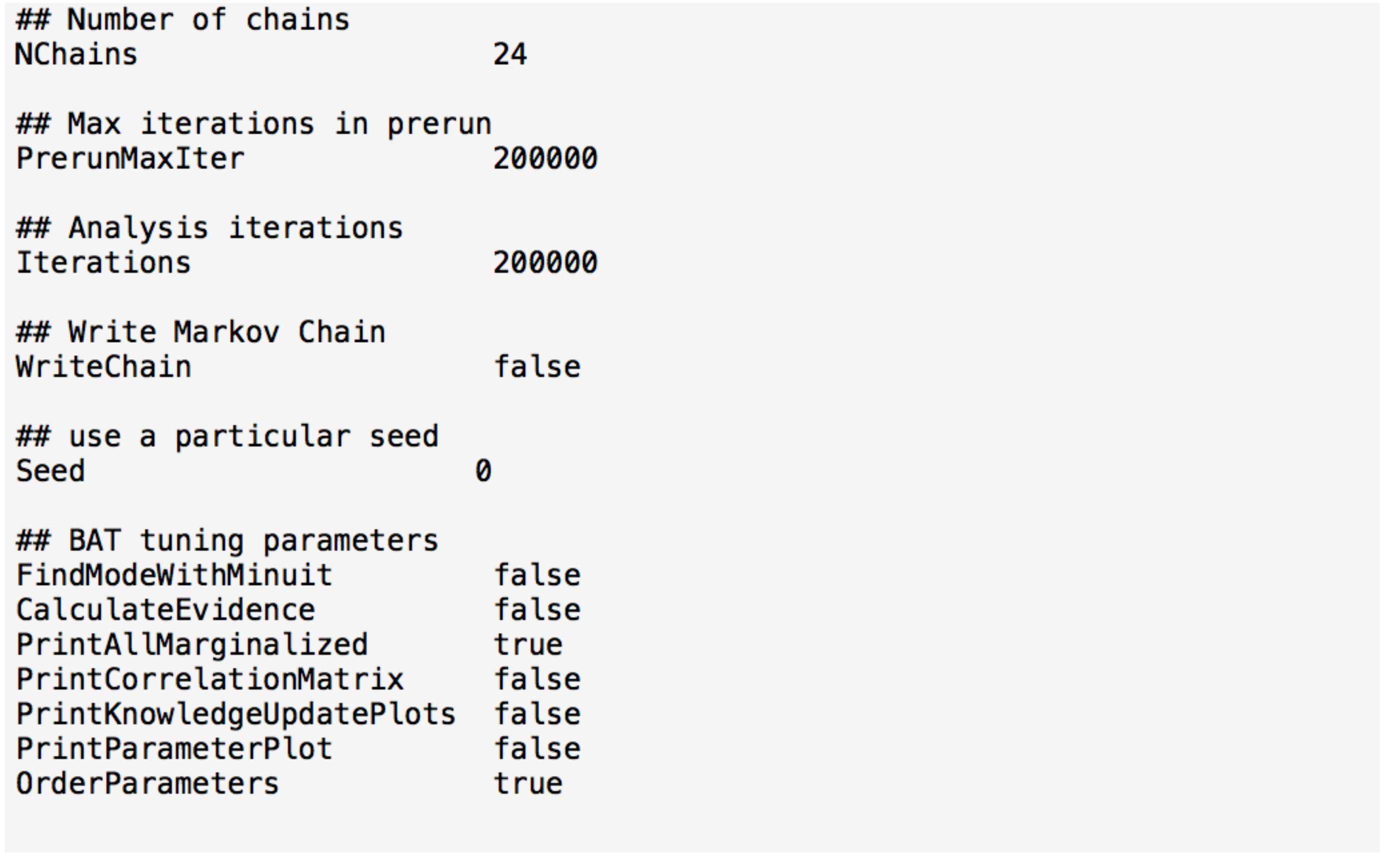
Parametric correlations set through the CorrelatedGaussianObservables method



```
######################################################################
CorrelatedGaussianObservables   LatticeV 2
Observable  a_0V    a_0V    a_0V    1   -1    MCMC weight  0.4975  0.0667   0.
Observable  a_1V    a_1V    a_1V    1   -1    MCMC weight -2.0151  0.9165   0.
1.          0.859005
0.859005    1.
#
CorrelatedGaussianObservables   LatticeA0_A12   4
Observable  a_0A0   a_0A0   a_0A0   1   -1    MCMC weight  0.5023  0.0370   0.
Observable  a_1A0   a_1A0   a_1A0   1   -1    MCMC weight -1.6084  0.4473   0.
Observable  a_0A12  a_0A12  a_0A12  1   -1    MCMC weight  0.2196  0.0238   0.
Observable  a_1A12  a_1A12  a_1A12  1   -1    MCMC weight  0.3324  0.3002   0.
1.          0.667316    0.904271    0.887787
0.667316    1.          0.909064    0.927202
0.904271    0.909064    1.          0.97564
0.887787    0.927202    0.97564     1.
#
CorrelatedGaussianObservables   LatticeA1 2
Observable  a_0A1   a_0A1   a_0A1   1   -1    MCMC weight  0.2848  0.0233   0.
Observable  a_1A1   a_1A1   a_1A1   1   -1    MCMC weight  0.1914  0.2804   0.
1.          0.948261
0.948261    1.
#
CorrelatedGaussianObservables   LatticeT1_T2 4
Observable  a_0T1   a_0T1   a_0T1   1   -1    MCMC weight  0.4197  0.0241   0.
Observable  a_1T1   a_1T1   a_1T1   1   -1    MCMC weight -1.3633  0.2586   0.
Observable  a_0T2   a_0T2   a_0T2   1   -1    MCMC weight  0.27997 0.01948  0.
Observable  a_1T2   a_1T2   a_1T2   1   -1    MCMC weight  0.1171  0.2364   0.
1.          0.500481    0.850285    0.820455
0.500481    1.          0.801509    0.836394
0.850285    0.801509    1.          0.93236
0.820455    0.836394    0.93236     1.
#
CorrelatedGaussianObservables   LatticeT23 2
Observable  a_0T23  a_0T23  a_0T23  1   -1    MCMC weight  0.5235  0.0451   0.
Observable  a_1T23  a_1T23  a_1T23  1   -1    MCMC weight -0.2714  0.5791   0.
1.          0.951964
0.951964    1.
#
```

the structure of the code

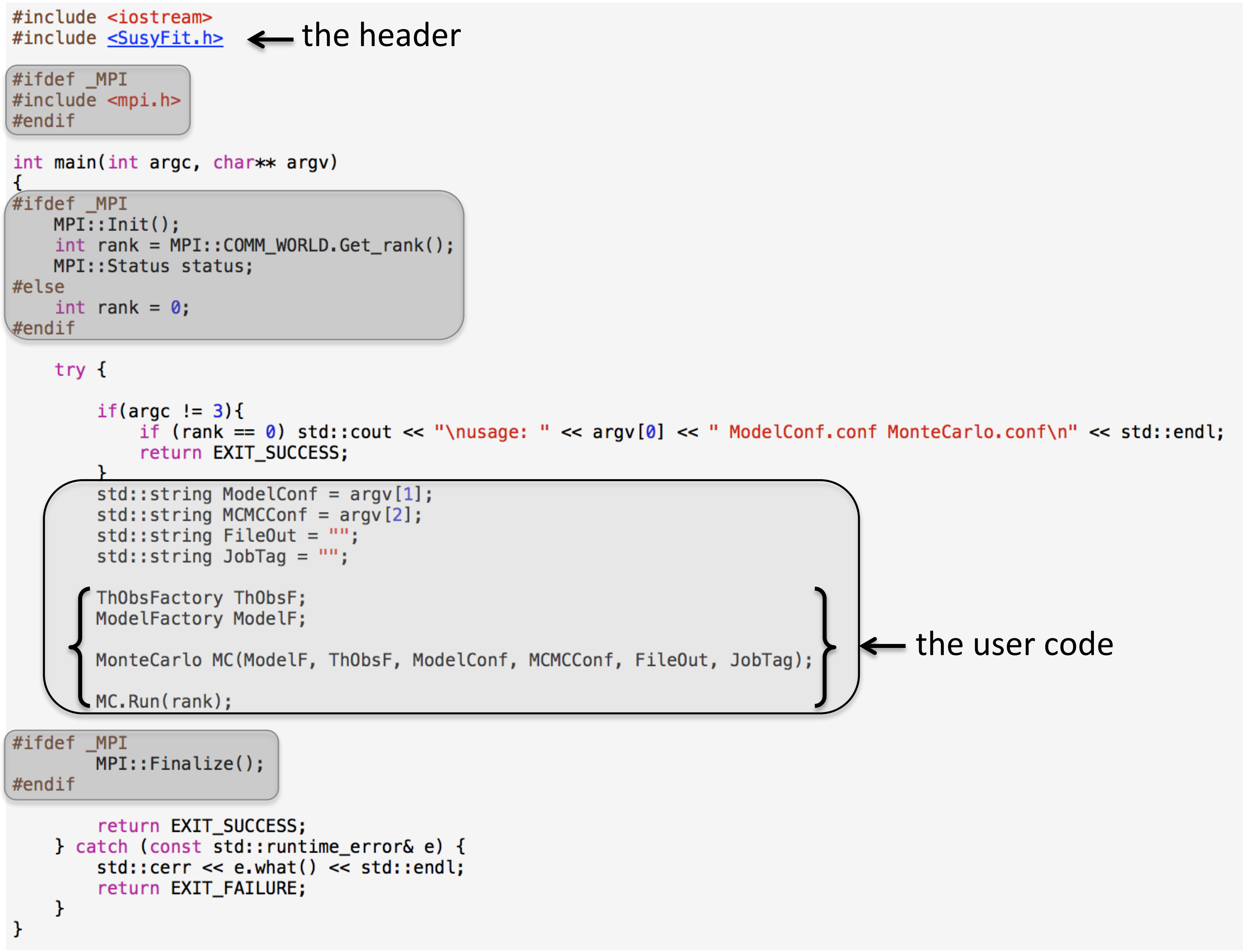
all MCMC parameters set through a separate configuration file.



```
## Number of chains
NChains                    24

## Max iterations in prerun
PrerunMaxIter              200000

## Analysis iterations
Iterations                 200000

## Write Markov Chain
WriteChain                 false

## use a particular seed
Seed                      0

## BAT tuning parameters
FindModeWithMinuit         false
CalculateEvidence          false
PrintAllMarginalized       true
PrintCorrelationMatrix     false
PrintKnowledgeUpdatePlots  false
PrintParameterPlot         false
OrderParameters            true
```

need help getting the code running? email me: apaul2@alumni.nd.edu

sample user codes for MCMC

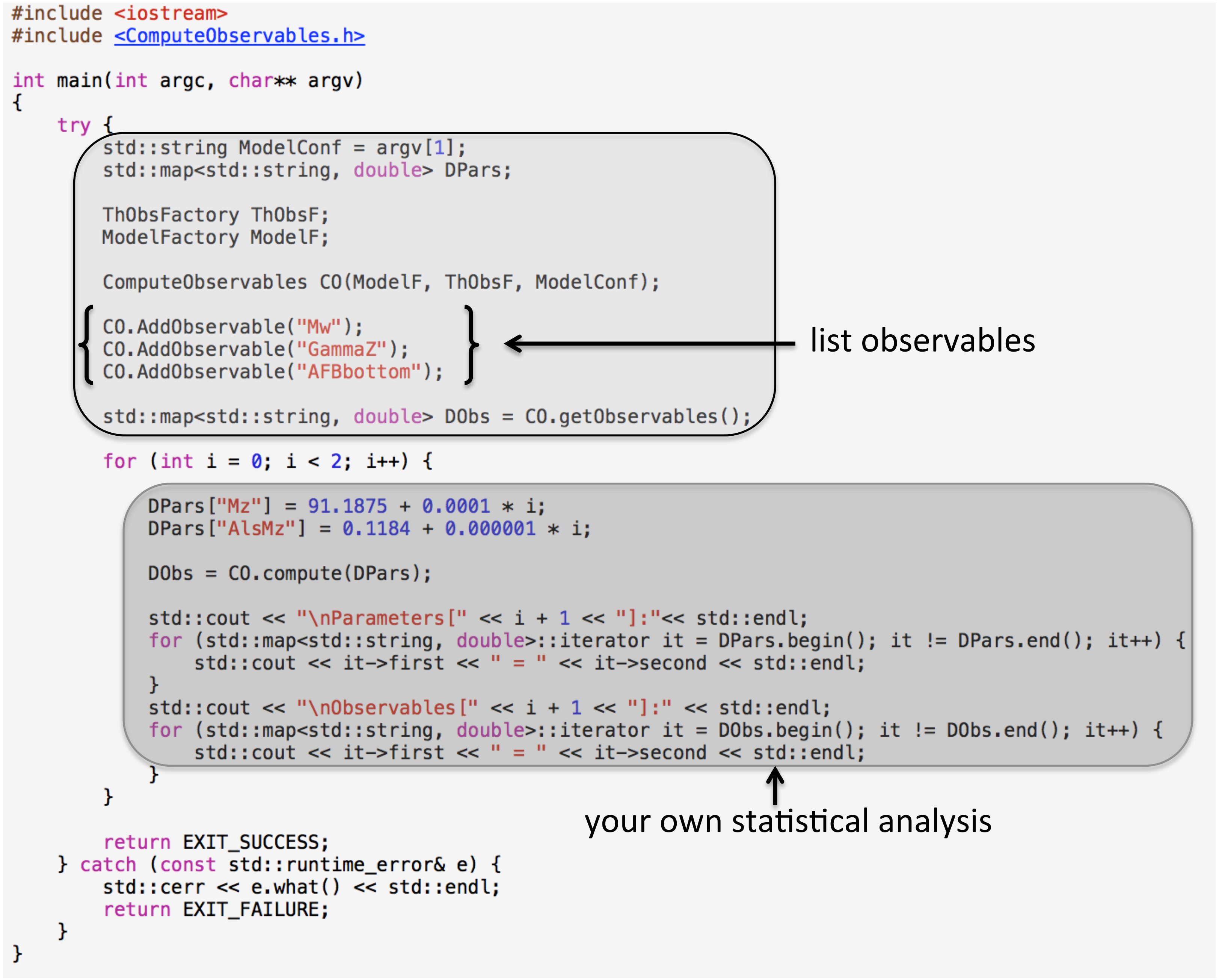


```
#include <iostream>
#include <SusyFit.h>

#ifdef _MPI
#include <mpi.h>
#endif

int main(int argc, char** argv)
{
#ifdef _MPI
    MPI::Init();
    int rank = MPI::COMM_WORLD.Get_rank();
    MPI::Status status;
#else
    int rank = 0;
#endif

    try {

        if(argc != 3){
            if (rank == 0) std::cout << "\nusage: " << argv[0] << " ModelConf.conf MonteCarlo.conf\n" << std::endl;
            return EXIT_SUCCESS;
        }
        std::string ModelConf = argv[1];
        std::string MCMCConf = argv[2];
        std::string FileOut = "";
        std::string JobTag = "";

        ThObsFactory ThObsF;
        ModelFactory ModelF;

        MonteCarlo MC(ModelF, ThObsF, ModelConf, MCMCConf, FileOut, JobTag);

        MC.Run(rank);

#ifdef _MPI
        MPI::Finalize();
#endif

        return EXIT_SUCCESS;
    } catch (const std::runtime_error& e) {
        std::cerr << e.what() << std::endl;
        return EXIT_FAILURE;
    }
}
```

← the header

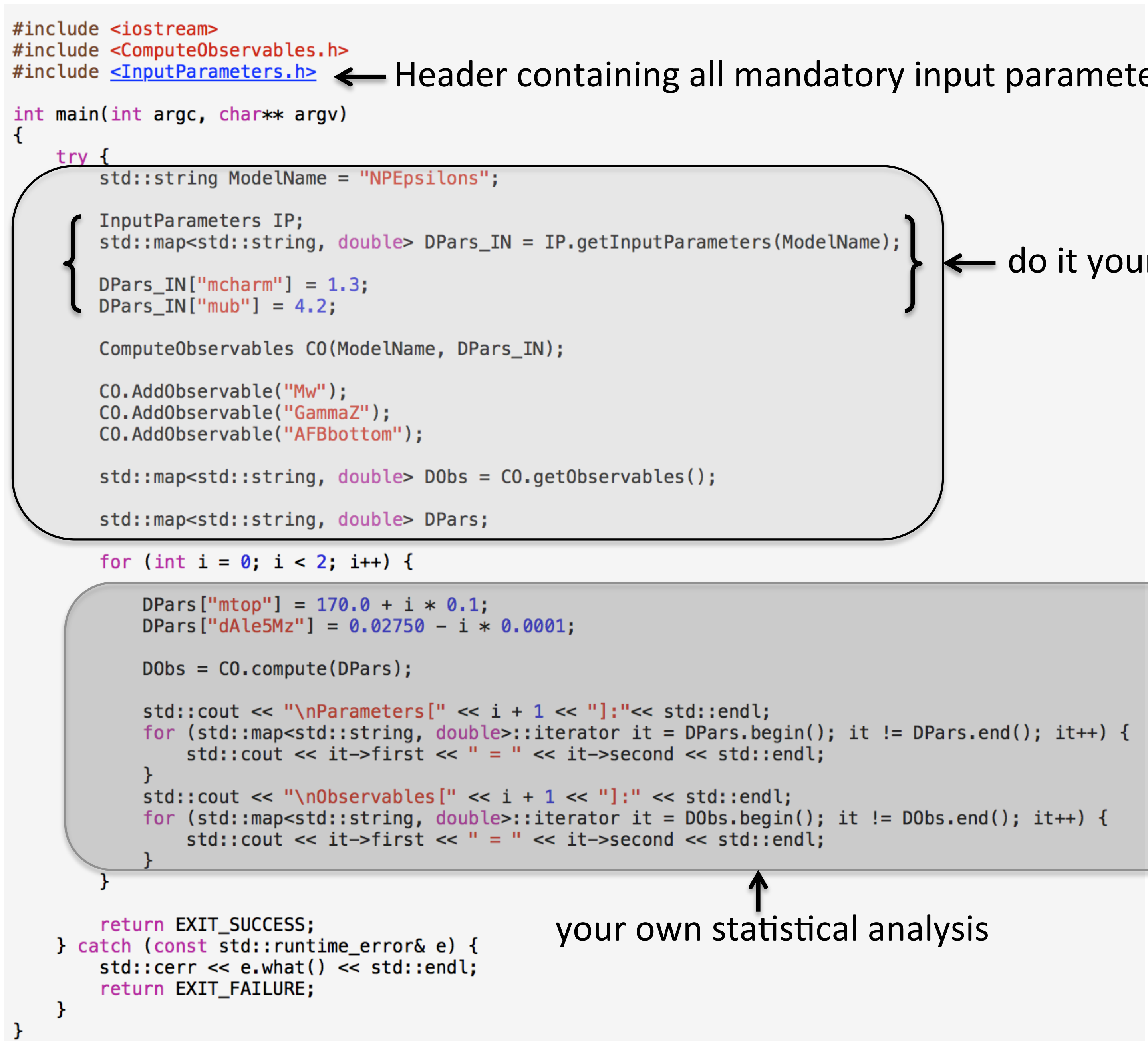
← the user code

to implement your own statistical analysis



```
#include <iostream>
#include <ComputeObservables.h>

int main(int argc, char** argv)
{
    try {
        std::string ModelConf = argv[1];
        std::map<std::string, double> DPars;

        ThObsFactory ThObsF;
        ModelFactory ModelF;

        ComputeObservables CO(ModelF, ThObsF, ModelConf);

        CO.AddObservable("Mw");
        CO.AddObservable("GammaZ");
        CO.AddObservable("AFBbottom");

        std::map<std::string, double> DObs = CO.getObservables();

        for (int i = 0; i < 2; i++) {

            DPars["Mz"] = 91.1875 + 0.0001 * i;
            DPars["AlsMz"] = 0.1184 + 0.000001 * i;

            DObs = CO.compute(DPars);

            std::cout << "\nParameters[" << i + 1 << "]:"<< std::endl;
            for (std::map<std::string, double>::iterator it = DPars.begin(); it != DPars.end(); it++) {
                std::cout << it->first << " = " << it->second << std::endl;
            }
            std::cout << "\nObservables[" << i + 1 << "]:" << std::endl;
            for (std::map<std::string, double>::iterator it = DObs.begin(); it != DObs.end(); it++) {
                std::cout << it->first << " = " << it->second << std::endl;
            }
        }

        return EXIT_SUCCESS;
    } catch (const std::runtime_error& e) {
        std::cerr << e.what() << std::endl;
        return EXIT_FAILURE;
    }
}
```

an option for the hardcore



```cpp
#include <iostream>
#include <ComputeObservables.h>
#include <InputParameters.h>

int main(int argc, char** argv)
{
    try {
        std::string ModelName = "NPEpsilons";

        InputParameters IP;
        std::map<std::string, double> DPars_IN = IP.getInputParameters(ModelName);

        DPars_IN["mcharm"] = 1.3;
        DPars_IN["mub"] = 4.2;

        ComputeObservables CO(ModelName, DPars_IN);

        CO.AddObservable("Mw");
        CO.AddObservable("GammaZ");
        CO.AddObservable("AFBbottom");

        std::map<std::string, double> DObs = CO.getObservables();

        std::map<std::string, double> DPars;

        for (int i = 0; i < 2; i++) {

            DPars["mtop"] = 170.0 + i * 0.1;
            DPars["dAle5Mz"] = 0.02750 - i * 0.0001;

            DObs = CO.compute(DPars);

            std::cout << "\nParameters[" << i + 1 << "]:"<< std::endl;
            for (std::map<std::string, double>::iterator it = DPars.begin(); it != DPars.end(); it++) {
                std::cout << it->first << " = " << it->second << std::endl;
            }
            std::cout << "\nObservables[" << i + 1 << "]:" << std::endl;
            for (std::map<std::string, double>::iterator it = DObs.begin(); it != DObs.end(); it++) {
                std::cout << it->first << " = " << it->second << std::endl;
            }
        }

        return EXIT_SUCCESS;
    } catch (const std::runtime_error& e) {
        std::cerr << e.what() << std::endl;
        return EXIT_FAILURE;
    }
}
```

$\leftarrow$ Header containing all mandatory input parameters

$\leftarrow$ do it yourself…

$\uparrow$ your own statistical analysis

nail-biting time?

- observable computation rate of O(kHz) or better for the implemented observables
- a simple test analysis can be done in minutes (statistics of O(10k))
- a results analysis can be done in hours (statistics of O(100k))
- a publication level analysis can be done in days (statistics of O(1M) or better)
- times are for single thread (core) single chain analyses on a laptop/desktop
- using the built in MPI implementation reduces time or increases statistics by a factor of O(N) for N threads (cores) used

caveat: computers only get faster…

➡ made a mistake? the code exits with a message telling you why.

caveat: no code is seg fault proof, however, error handling is one of our priorities

want someone to talk to while waiting? email me: apaul2@alumni.nd.edu

model menu

- **Standard Model** (fully tested and results available in the literature with more to come soon)

- **general MSSM** (including variants like MFV, pMSSM etc.) with SLHA2 compatibility
  FeynHiggs to be used as the spectrum generator

- **two Higgs doublet models** (under construction)

- **some model independent extensions** for the study of NP in EW and Higgs physics (dim-6 operators, oblique parameters etc. tested)

to write a custom model one can use the template that we will provide, adding the necessary input and derived parameters

custom observables can also be added which can depend on the parameters of the existing models and/or the custom model

observables menu

- **EW precision observables** (tested)

$$M_W,\ \Gamma_W,\ \Gamma_Z,\ \sigma_h^0,\ \sin^2\theta_{\rm eff}^{\rm lept}(Q_{\rm FB}^{\rm had}),\ P_\tau^{\rm pol},\ \mathcal{A}_f,\ A_{\rm FB}^{0,f},\ R_f^0$$
$$\text{for}\ \ f=\ell,c,b$$

- **Higgs boson signal strengths** (tested)

$$H\to\gamma\gamma,\ ZZ,\ WW,\ \tau^+\tau^-,\ b\bar{b}\ \ \text{for different categories}$$

- **Lep2 two fermion processes** (tested)

$$\sigma\ \text{and}\ A_{\rm FB}\ \ \text{for}\ e^+e^-\to e^+e^-,\ \mu^+\mu^-,\ \tau^+\tau^-,\ c\bar{c},\ b\bar{b}$$

observables menu

- **Unitarity triangle observables** (tested against UTFit)

  UT angles, $\Delta F = 2$ amplitudes, CKM elements

- **rare decays** (under development)

  $B \to X_s \gamma,\ B \to K^* \gamma$ (in progress)

  $B \to X_s \ell^+ \ell^-,\ B \to K \ell^+ \ell^-$ (in progress)

  $B \to K^* \ell^+ \ell^-$

  $B_{s,d} \to \mu^+ \mu^-$

  $K \to \pi \nu \bar{\nu}$ (in progress)

  $K \to \mu^+ \mu^-$ (in progress)

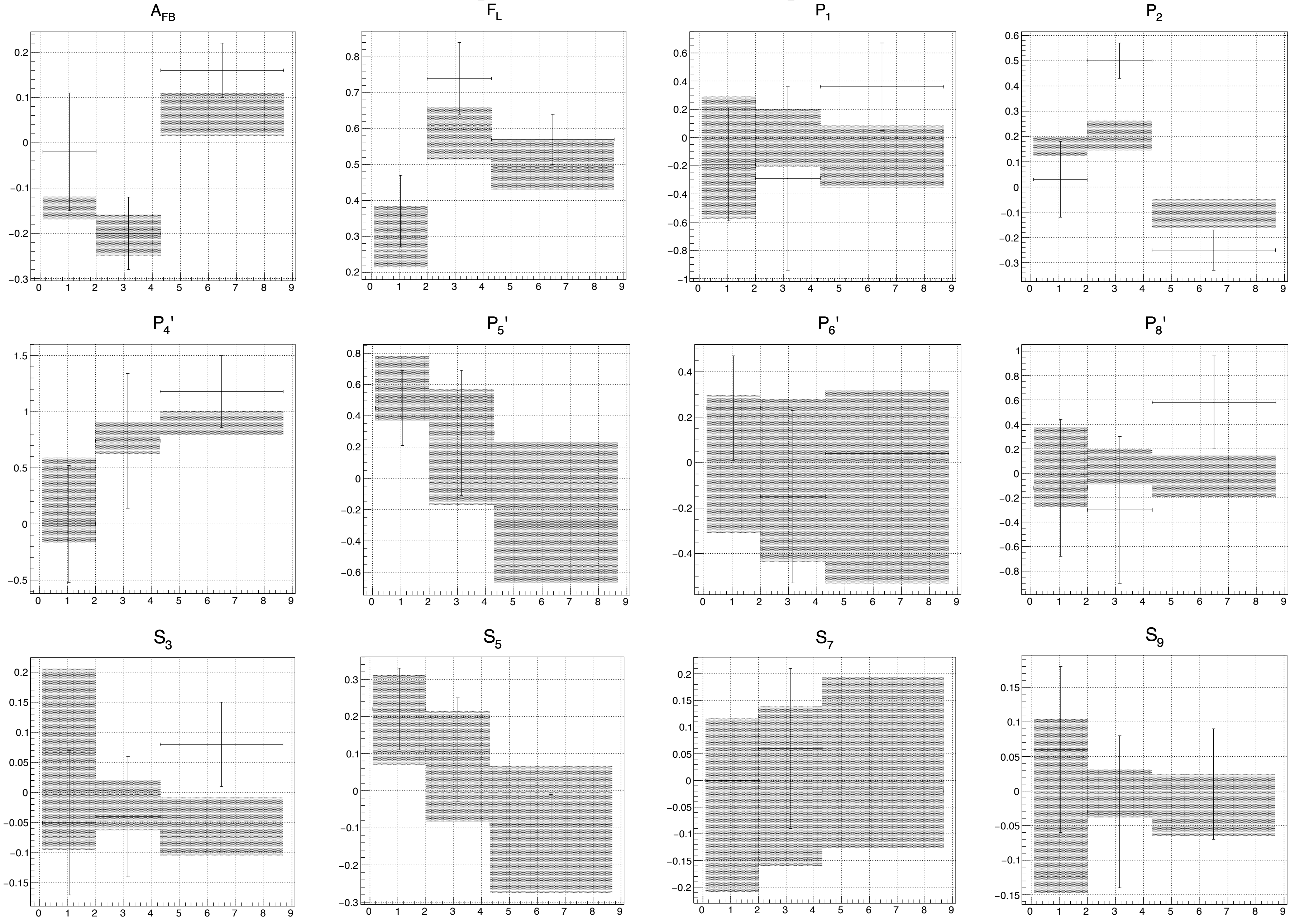
  $\tau \to \mu \gamma,\ \tau \to 3\ell$ (+other LFV processes, in progress)

- **non-leptonic decays** (tested)

  $B \to PP,\ PV$ (in progress)

  ϵ' / ϵ (in progress)

proof of concept



utility and prospects

- **theory predictions**:

this framework can simply be used for theory predictions of observables for a certain set or a range of the set of parameters

- **phenomenological studies**:

this can also be used for detailed phenomenological studies under a chosen statistical framework

- **correlations**:

this can be used for studies of correlations between different observables within a given parameter space of a given model

- **theoretical uncertainties**:

a very effective tool for deriving parametric uncertainties coming from the whole or part of the theory parameter space

summary

- ✓ we are developing a computational framework that we hope will be of much utility to both theorist and experimentalists

- ✓ a tool that will combine direct and indirect searches

- ✓ it will allow for easy comparisons and phenomenological analyses of experimental results in the light of theoretical frameworks within the Standard Model and beyond

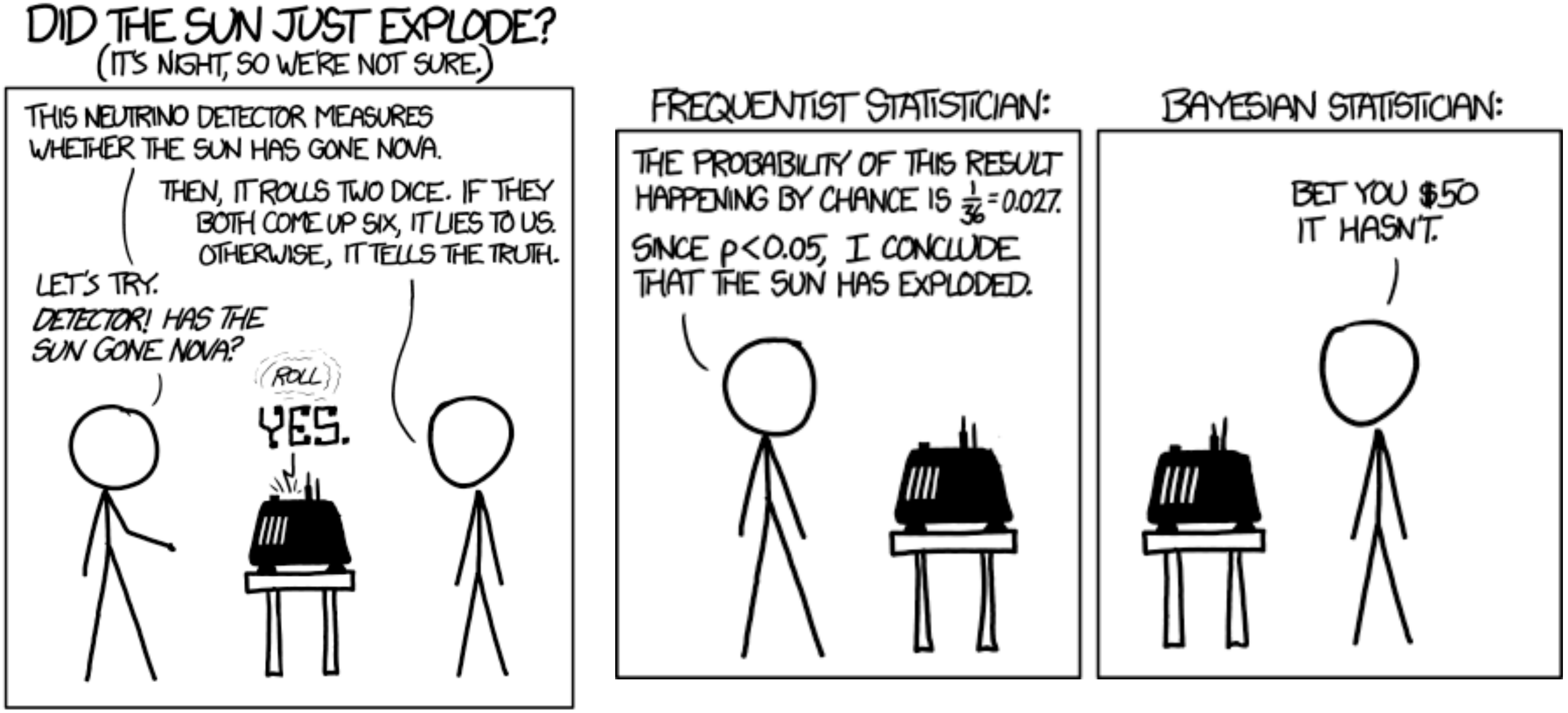
- ✓ a flexible framework that allows for different kinds of statistical analyses with a Bayesian analysis built in based on very efficient and fast Markov Chain Monte Carlo routines

- ✓ a modular structure that allows for users to modify or add to the existing one

- ✓ uses the leading industry standards in MPI, GSL, Boost and ROOT

- ✓ will come with full documentation in the form of code documentation (doxygen) and an "instruction manual" for users including physics references

the usual debate…


DID THE SUN JUST EXPLODE?
(IT'S NIGHT, SO WE'RE NOT SURE.)
THIS NEUTRINO DETECTOR MEASURES WHETHER THE SUN HAS GONE NOVA.
THEN, IT ROLLS TWO DICE. IF THEY BOTH COME UP SIX, IT LIES TO US. OTHERWISE, IT TELLS THE TRUTH.
LET'S TRY. DETECTOR! HAS THE SUN GONE NOVA?
ROLL
YES.
FREQUENTIST STATISTICIAN:
THE PROBABILITY OF THIS RESULT HAPPENING BY CHANCE IS 1/36 = 0.027. SINCE p<0.05, I CONCLUDE THAT THE SUN HAS EXPLODED.
BAYESIAN STATISTICIAN:
BET YOU $50 IT HASN'T.

now you have options…

the developers

Roma:
Shehu S. AbdusSalam
Jorge de Blas
Debtosh Chowdhury
Otto Eberhardt
Marco Fedele
Enrico Franco
Diptimoy Ghosh
Ayan Paul
Luca Silvestrini

Roma Tre:
Marco Ciuchini

KIAS:
Satoshi Mishima

ICTP/SISSA:
Giovanni Grilli di Cortona
Ivan Girardi
Mauro Valli

Florida State U.:
Laura Reina

Caltech:
Maurizio Pierini (CMS)

We look forward to welcoming developers willing to add models/observables to the existing framework and/or run tests against codes existing in the market.

in a few decades we will put an artificial intelligence engine in the code…

Thank you…!!

To my Mother and Father, who showed me what I could do,

and to Ikaros, who showed me what I could not.

“To know what no one else does, what a pleasure it can be!”

– adopted from the words of

Eugene Wigner.